WHAT IS CLAIMED IS:

1. A task allocation method of allocating a task selectively to a first processor and a second processor in a multiprocessor system, the first processor having a first instruction set and the second processor a second instruction set, and the task corresponding to a program having an execution efficiency, the method comprising:

allocating a task corresponding to a program module described by the first instruction set to the first processor;

determining whether or not the execution efficiency of the program is improved if a destination allocated for the task is changed from the first processor to the second processor; and

changing the destination allocated for the task to the second processor if the execution efficiency of the program is improved.

2. The method according to claim 1, further comprising:

estimating a first execution time of the program in a case where the task is allocated to the first processor and estimating a second execution time of the program in a case where the task is allocated to the second processor; and

determining whether the second execution time is shorter than the first execution time, in order to determine whether or not the execution efficiency of the program is improved.

3. The method according to claim 1, further comprising:

estimating a first data amount processible by the task within a unit time in a case where the task is allocated to the first processor and estimating a second data amount processible by the task within a unit time in a case where the task is allocated to the second processor; and

determining whether the second data amount is larger than the first data amount, in order to determine whether or not the execution efficiency of the program is improved.

4. The method according to claim 1, further comprising:

estimating a first data amount processible by the task within a unit time in a case where the task is allocated to the first processor and estimating a second data amount processible by the task within a unit time in a case where the task is allocated to the second processor;

estimating an increment of data amount between the first data amount and the second data amount; and

determining whether the increment is larger than a preset threshold, in order to determine whether or not the execution efficiency of the program is improved.

5. The method according to claim 1, further comprising:

estimating a load on of the second processor in a case where the destination allocated for the task is changed from the first processor to the second processor;

determining whether the load on the second processor is an overload, in order to determine whether or not the execution efficiency of the program is improved.

6. The method according to claim 1, further comprising:

estimating a first amount of data transferred by inter-processor communication in the program in a case where the task is allocated to the first processor and estimating a second amount of data transferred by inter-processor communication in the program in a case where the task is allocated to the second processor; and

determining whether the second amount of data is smaller than the first amount of data, in order to determine whether or not the execution efficiency of the program is improved.

7. The method according to claim 1, further comprising:

estimating a first amount of data transferred by inter-processor communication within a unit time in the program in a case where the task is allocated to the first processor and estimating a second amount of data transferred by inter-processor communication within a unit time in the program in a case where the task is allocated to the second processor; and

determining whether the second amount of data within the unit time is smaller than the first amount of data within the unit time, in order to determine whether or not the execution efficiency of the program is improved.

8. The method according to claim 1, further comprising:

acquiring a program module described by the second instruction set, which is necessary for creating a task to be allocated to the second processor, by replacing a first instruction, in the program module described by the first instruction set, with a second instruction of the second instruction set, for executing the same process as the first instruction.

9. The method according to claim 1, further comprising:

acquiring a program module described by the second instruction set, which is necessary for creating a task to be allocated to the second processor, by compiling a source code for the program module described by the first instruction set with a compiler for the second processor.

10. The method according to claim 1, further comprising:

acquiring a program module described by the second instruction set from a file system or a network.

11. The method according to claim 1, wherein the program module is a program module complex including a first program module described by the first instruction set for the first processor and a second program module described by the second instruction set for the second processor, and wherein said task is created by using one of the first program module and the second program module.

12. The method according to claim 1, further comprising:

updating a task allocation table storing task allocation information, in response to the changing of the destination allocated for the task to the second processor.

13. A multiprocessor system having a first processor with a first instruction set and a second processor with a second instruction set, the system comprising:

a task allocation unit configured to allocate a task that corresponds to a program including a program module described by the first instruction set to the first processor, the program having an execution efficiency;

a determination unit configured to determine whether or not the execution efficiency of the program is improved if a destination allocated for the task is changed from the first processor to the second processor; and

a task allocation control unit configured to change the destination allocated for the task to the second processor if the execution efficiency of the program is improved.

14. The system according to claim 13, further comprising:

an estimation unit configured to estimate a first execution time of the program in a case where the task is allocated to the first processor and to estimate a second execution time of the program in a case where the task is allocated to the second processor; and

a determination unit configured to determine whether the second execution time is shorter than the first execution time, in order to determine whether or not the execution efficiency of the program is improved.

15. The system according to claim 13, further comprising:

an estimation unit configured to estimate a first data amount processible by the task within a unit time in a case where the task is allocated to the first processor and to estimate a second data amount processible by the task within a unit time in a case where the task is allocated to the second processor; and

a determination unit configured to determine whether the second data amount is larger than the first data amount, in order to determine whether or not the execution efficiency of the program is improved.

16. The system according to claim 13, further comprising:

an estimation unit configured to estimate a first data amount processible by the task within a unit time in a case where the task is allocated to the first processor and to estimate a second data amount processible by the task within a unit time in a case where the task is allocated to the second processor, thereby estimating an increment of data amount between the first data amount and the second data amount; and

a determination unit configured to determine whether the increment of data amount is larger than a preset threshold, in order to determine whether or not the execution efficiency of the program is improved.

17. The system according to claim 13, further comprising:

an estimation unit configured to estimate a load on of the second processor in a case where the destination allocated for the task is changed from the first processor to the second processor;

a determination unit configured to determine whether the load on the second processor is an overload, in order to determine whether or not the execution efficiency of the program is improved.

18. The system according to claim 13, further comprising:

an estimation unit configured to estimate a first amount of data transferred by inter-processor communication in the program in a case where the task is allocated to the first processor and to estimate a second amount of data transferred by inter-processor communication in the program in a case where the task is allocated to the second processor; and

a determination unit configured to determine whether the second amount of data is smaller than the first amount of data, in order to determine whether or not the execution efficiency of the program is improved.

19. The system according to claim 13, further comprising:

an estimation unit configured to estimate a first amount of data transferred by inter-processor communication within a unit time in the program in a case where the task is allocated to the first processor and to estimate a second amount of data transferred by inter-processor communication within a unit time in the program in a case where the task is allocated to the second processor; and

a determination unit configured to determine whether the second amount of data within the unit time is smaller than the first amount of data within the unit time, in order to determine whether or not the execution efficiency of the program is improved.

20. The system according to claim 13, further comprising:

an acquisition unit configured to acquire a program module described by the second instruction set, which is necessary for creating a task to be allocated to the second processor, by replacing a first instruction, in the program module described by the first instruction set, with a second instruction of the second instruction set, for executing the same process as the first instruction.

21. The system according to claim 13, further comprising:

an acquisition unit configured to acquire a program module described by the second instruction set, which is necessary for creating a task to be allocated to the second processor, by compiling a source code for the program module described by the first instruction set with a compiler for the second processor.

22. The system according to claim 13, further comprising:

an acquisition unit configured to acquire a program module described by the second instruction set from a file system or a network.

23. The system according to claim 13, wherein the program module is a program module complex including a first program module described by the first instruction set for the first processor and a second program module described by the second instruction set for the second processor, and wherein said task is created by using one of the first program module and the second program module.

24. The system according to claim 13, further comprising:

a task allocation table storing task allocation information, in response to the changing of the destination allocated for the task to the second processor.

25. A program product comprising a computer usable medium having computer readable program code means for causing a multiprocessor system having a first processor with a first instruction set and a second processor with a second instruction set, to allocate a task to either of the first processor or the second processor, the task corresponding to a program having an execution efficiency, the program including a program module described by either of the first instruction set or the second instruction set, the computer readable program code means in the computer

program product comprising:

program code means for causing the multiprocessor system to allocate a task that corresponds to a program module described by the first instruction set to the first processor;

program code means for causing the multiprocessor system to determine whether or not the execution efficiency of the program is improved if a destination allocated for the task is changed from the first processor to the second processor; and

program code means for causing the multiprocessor system to change the destination allocated for the task to the second processor if the execution efficiency of the program is improved.